VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ PORTÁL DOPLATKŮ LÉKŮ A LÉKÁREN

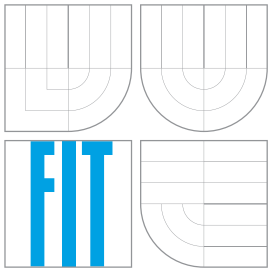
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DANIEL KODEŠ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ PORTÁL DOPLATKŮ LÉKŮ A LÉKÁREN

WEB APPLICATION FOR THE ADDITIONAL MEDICAL PAYMENTS, AND CHEMISTS' DATABASE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DANIEL KODEŠ
AUTHOR

VEDOUCÍ PRÁCE — Ing. RUTTKAY LADISLAV,
SUPERVISOR

BRNO 2009

Abstrakt

Bakalářská práce popisuje analýzu, návrh a implementaci webového portálu doplatků léků a lékáren. Čtenář by měl po přečtení pochopit, jak se tvoří ceny léků v České republice a tím i samotný význam portálu. Návrh portálu byl popsán jazykem UML. Práce se podrobněji věnuje implementaci portálu pomocí technologie ASP.NET. Výsledný portál umožní uživateli nalézt nejlevnější cenu léku v jeho okolí, včetně adresy a otevírací doby lékárny, která lék prodává.

Abstract

This bachelor's thesis describes analysis, design and implementation of web application for the additional medical payments, and chemist's database. A reader should after reading understand, how is medicament price made up in Czech republic and by this an application meaning. The application's design was described with UML. The thesis is closely focused on the implementation of portal using ASP. NET technology. Final application allow user find lowest medicament's price in his neighbourhood, including chemist's address and opening hours, which sells the medicament.

Klíčová slova

.NET, ASP.NET, C#, MS SQL, Microsoft, web, portal, AJAX

Keywords

.NET, ASP.NET, C#, MS SQL, Microsoft, web, portal, AJAX

Citace

Daniel Kodeš: Webový portál doplatků léků a lékáren, bakalářská práce, Brno, FIT VUT v Brně, 2009

Webový portál doplatků léků a lékáren

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ladislava Ruttkaye

.....
Daniel Kodeš
20. května 2009

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce bylo vytvořit webový portál, který usnadní uživatelům nalézt nejlevnější lék v jejich bydlišti nebo vybrané lokaci.

První kapitola obsahuje základní informace o tvorbě cen léků v České republice a úskalí s tím spojená. Je zde také prezentován výsledek průzkumu o cenách léků v jednotlivých lékárnách.

Cílem druhé kapitoly je popsat použité technologie. O každé technologii je zde její popis a využití v systému.

Třetí kapitola zobrazuje analýzu a návrh systému. Popisuje, s využitím různých technik, chování systému a konceptuální model.

Ve čtvrté kapitole je popsána samotná implementace webového portálu. Popisuje se zde použitá architektura aplikace, uživatelské účty a jednotlivé části portálu.

Závěrečná pátá kapitola obsahuje zhodnocení výsledné práce a navrhuje možná budoucí vylepšení portálu.

1.1 Obecné informace k lékům

Cena léků není pevně daná, ale ovlivňuje jí několik faktorů. V důsledku toho dochází v jednotlivých lékárnách k rozdílným cenám téhož léku. U dražších léků tento rozdíl už může znamenat nezanedbatelnou částku.

Od roku 2008 se za léky, které jsou vydávány na předpis, hradí navíc regulační poplatek 30 Kč. U některých levných léků tento poplatek znamená celou cenu léku nebo ještě více. Tato nepříjemnost se obchází tím, že lékař vyznačí na předpis „hradí pacient“.

1.2 Tvorba cen léků

U každého léku, buď od tuzemského výrobce nebo dovezeného ze zahraničí, se stanovuje maximální cena, za kterou se smí lék prodávat. Tuto maximální cenu stanovuje Státní ústav pro kontrolu léčiv [6]. Lékárna však může tuto maximální cenu snížit. Dále Ministerstvo zdravotnictví stanovuje maximální ceny za výkony obchodu (tj. maximální obchodní přirážka). Tato cena se vypočítává z procentuální sazby a ceny. Procentuální sazba se určuje podle pásma, dle ceny. Pásem je celkem 8. Pokud se obchodu účastní více stran, tak součet jejich výkonů nesmí překročit maximalní obchodní přirážku. Následně se od vypočtené částky odečítá částka hrazená pojišťovnou. Výsledná cena nesmí být nižší než 0 Kč. Pojišťovna hradí pouze léky, které jsou na předpis. K výsledné ceně se připočte DPH 9%. Kompletní přepočet cen se provádí obvykle jedenkrát za kvartál.

Velké lékárny, které mají vysoký odběr léků, mívají většinou nižší ceny, které si mohou dovolit, protože dostávají od dodavatelů množstevní slevy. Menší lékárny se tomuto snaží konkurovat snížením svých marží.

1.3 Průzkum

Za účelem získání přehledu o rozdílných cenách v jednotlivých lékárnách jsem provedl průzkum v několika vybraných lékárnách v Českých Budějovicích. Snažil jsem se vybrat lékárny, které si nejsou podobné svou lokací nebo velikostí. Pro porovnání výsledků jsem vybral léky z různých cenových hladin.

Průzkum byl proveden dne 29. 1. 2009. Ceny jsou uvedeny v Kč a jedná se pouze o doplatky léků.

	Lékárna 1	**Lékárna 2**	**Lékárna 3**
Zyrtec 20 × 10 mg	53.8	63	59
Rowatinex kapky	122	128	132
Condrosulf 400 60 × 400 mg	156	123	141
Gingio 80 120 × 80 mg	398	415	421

Tabulka 1.1: Přehled cen léků v jednotlivých lékárnách.

V tabulce je vidět, že ceny léků se v jednotlivých lékárnách opravdu liší, ale není pevně dané, že jedna lékárna má vždy nejnižší ceny.

Kapitola 2

Použité technologie

Tato kapitola obsahuje popis jednotlivých technologiích použitých v systému. Ukazuje prostředky, které se využívají v analýze a návrhu, či až v samotné implementaci.

2.1 UML

UML (Unified Modeling Language) je grafický jazyk, který slouží pro vizualizaci, specifikaci, stavbu a dokumentaci softwarových systémů. První verze byla vytvořena firmou Rational v roce 1997. Postupným vývojem se došlo v roce 2004 k verzi 2.0. V současnosti se připravuje verze 2.1.

Umožnuje modelovat aplikace pomocí pevně dané syntaxe, proto není problém vytvořené modely sdílet s ostatními návrháři. UML je navrženo tak, aby modelům porozumněl i zadavatel, který není úplně technologicky zdatný. Tímto se zajistí, že případná nedorozumnění nebo nepochopení požadavků se odhalí již ve fázi analýzy a návrhu.

Více informací lze nalézt v literatuře [2].

2.1.1 Diagram případu užití (Use Case diagram)

Diagram případu užití modeluje interakci uživatelů se systémem, aby se lépe pochopily požadavky zadavatelů a přesně vymezil rozsah vytvářeného systému. To znamená, že systém bude obsahovat pouze to, co je uvedeno v případu užití.

Uživatelé komunikující se systémem se nazývají aktéři. Aktér definuje roli, v které uživatel vystupuje v rámci komunikace se systémem. Jeden fyzický uživatel může vystupovat v několika rolích. Aktérem muže být i externí systém nebo čas.

2.1.2 Konceptuální modelování

Konceptuální modelování vytváří model konceptů aplikace, se kterými vyvíjený systém pracuje. Důležitými koncepty a jejich vztahy jsou koncepty, které reprezentují data uložená v databázi. Konceptuálního modelování využívá i ER (Entity-relationship) diagram, který patří mezi nejznámější a nejvyužívanější techniku pro navrhování databází.

2.1.3 Diagram tříd (Class diagram)

Diagramy tříd zobrazují statickou stránku systému, především vztahy mezi třídami. UML rozlišuje několik druhů tříd a množství vztahů, které jednotlivé třídy propojují (asociace, agregace, kompozice,...). Diagram tříd by měl být ve finále ekvivalentní zdrojovému kódu.

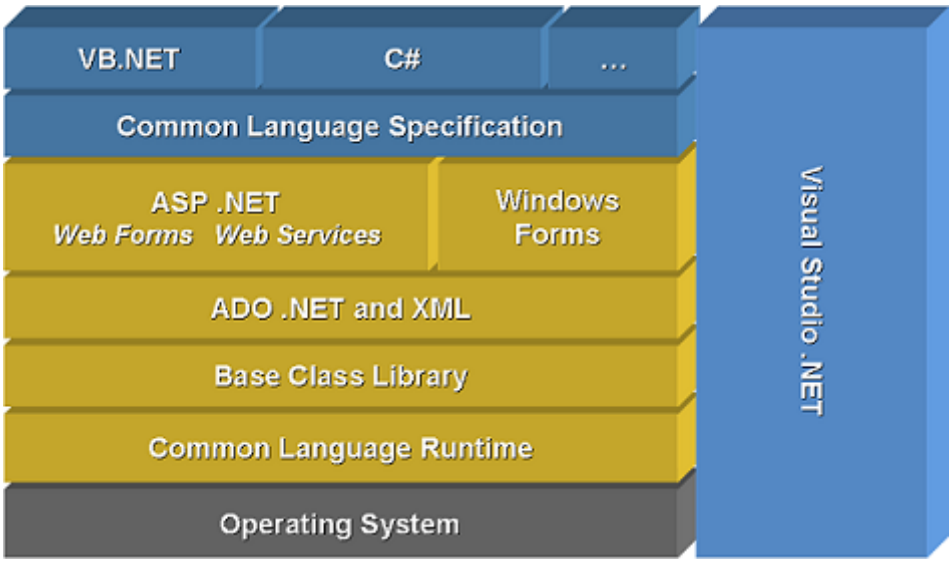
2.1.4 Diagram balíčků (Package diagram)

Diagram balíčků znázorňuje balíčky, které obsahují jednotlivé diagramy tříd. Pomocí šipek jsou mezi balíčky znázorněny závislosti mezi diagramy tříd. Balíčky jsou znázorněny jako složky, které v sobě drží třídy z diagramu tříd. Diagram balíčků se používá pro znázornění vrstev aplikační architektury.

2.2 Microsoft .NET Framework

Microsoft .NET Framework je softwarový framework pro některé operační systémy Windows. Díky open source projektu Mono je možné aplikace spouštět i na jiných platformách, jako je Linux nebo Mac. Obsahuje velkou knihovnu připravených řeseních obecných programových problémů. Tato knihovna se nazývá Base Class Library. Obsahuje podporu pro programování uživatelského prostředí, přístupu k datům, kryptografii, vytváření webových stránek a další.

Programy je možné psát ve vybraném programovacím jazyku, které .NET framework podporuje. Mezi jazyky patří VB.NET, C#, C++, JScript a J#. Tato nezávislost je zajištěna pomocí CLI (Common Language Infrastructure). Programy napsané ve vybraném jazyce se kompilují do přechodného jazyka CIL (Common Intermediate Language). Kód v tomto přechodném jazyku je vykonáván virtuálním strojem CLR (Common Language Runtime).



Obrázek 2.1: Architektura .NET Frameworku - zdroj [5]

2.2.1 ASP.NET

ASP.NET je nástroj pro vytváření dynamických webových aplikacích. Nahrazuje původní ASP (Active Server Page), které bylo intepretované. Stránky napsané v ASP.NET jsou kompilované, čímž se dosahuje vyššího výkonu než u stránek, které jsou intepretované. ASP.NET nabízí úplný, objektově orientovaný programovací model. Díky tomu, že je integrováné do .NET Frameworku je možné psát webové aplikace v oblíbeném programovacím jazyku, tím se také ulehčil přechod programátorům z desktopových aplikacích. Dále například nabízí podporu pro vícejazyčnost.

Poskytuje velké množství serverových ovládacích prvků, které usnadňují a zrychlují práci. Mezi ně patří ovládacích prvky pro zobrazení dat a jejich stránkování, dále prvky usnadňující validaci vstupních dat a prvky pro správu a autorizaci uživatelů. Nabízí také například Vzory stránek, díky kterým lze oddělit obsah od zbytku stránky. Značným vylepšením jsou také ovládací prvky pro technologii AJAX, která umožňuje asynchronní komunikaci se serverem v pozadí stránky.

V současnosti je ASP.NET ve verzi 3.5, která obsahuje nové technologie jako jsou LINQ (Language Integrated Query), zmíněný AJAX (Asynchronous JavaScript and XML) a Silverlight.

Více informací lze nalézt v literatuře [3].

2.2.2 Microsoft Visual Studio

Microsoft Visual Studio je integrované vývojové prostředí (IDE). Podporuje několik programovacích jazyků a je primárně určeno k vývoji aplikací pomocí .NET Frameworku. Je možné v něm vytvářet klasické Windows Forms a webové aplikace, webové služby i například aplikace pro mobilní zařízení.

Obsahuje v sobě editor kódu podporující IntelliSence (automatické doplňování) a zvýrazňování syntaxe. Dále má v sobě integrovaný debugger (nástroj pro ladění aplikací). Visual Studio taktéž obsahuje vizuální návrhový nástroj, který značně urychluje implementaci.

2.3 Microsoft SQL Server

Microsoft SQL Server je relační databázový systém, jehož dotazovacím jazykem je T-SQL. Podporuje uložené procedury a transakce. Zajišťuje integritu dat a jejich bezpečné uložení na serveru. Obsahuje pokročilé služby jako OLAP, Data mining, Reporting a další. V současnosti je ve verzi Microsoft SQL Server 2008. Pro komunikaci s Microsoft SQL Serverem je v .NET Frameworku komponenta ADO.NET.

Pro jednoduchou správu SQL serveru lze využít například Microsoft SQL Server Management Studio. To dodáva rohraní pro provádění operací, které vývojář nemusí umět zapsat v T-SQL nebo jejiž zápis by byl složitý. Pomocí tohoto nástroje lze jednoduše vytvářet tabulky, provádět import/export dat, zálohu a obnovu databáze a mnoho dalších užitečných funkcích.

2.4 jQuery

jQuery je knihovna pro javascript, která přináší mnoho funkcí na provádění operací, které by bylo v obyčejném javascriptu komplikované implementovat. Umožňuje manipulovat s DOM (Document Object Model) XHTML stránky, pracovat s CSS, provádění animací a mnoho

dalších funkcích. Při implementaci nemusí vývojář přemýšlet nad různým zpracováním jednotlivými prohlížečemi, protože knihovna si sama tento problém zpracuje. Velkou výhodou jQuery je také to, že už si kolem sebe vytvořila velkou komunitu vývojářů, kteří vytvářejí pluginy chybějící v jádře knihovny.

2.5 XHTML

XHTML (eXtensible HyperText Markup Language) je značkovací jazyk pro tvorbu webových stránek. Je následovníkem původního HTML s přidáním XML. Jazyk využívá tagy pro definování struktury textu, odstavců, tabulek atd. ve webové stránce.

Kód napsaný v XHTML je oproti kódu napsanému v HTML přehlednější a srozumitelnější. XHTML taky přináší některá nová pravidla pro psaní tagů, například všechny tagy musí být ukončeny, psány malými písmeny a hodnoty atributů musí být uzavřeny do uvozovek.

2.6 Kaskádové styly

Kaskádové styly CSS (Cascading Style Sheets) přinášejí podporu pro oddělení vzhledu stránky od kódu. To znamená, že pouhým vyměněním souboru stylů lze kompletně změnit vzhled stránky. Umožňuje psát jednoduchý a přehledný kód, který může definovat vzhled jediného elementu na stránce nebo vnořeného do jiného elementu. CSS navíc přináší nové možnosti změny zvhledu, které v kódu nebyly možné. Nevýhodou je však špatná podpora některých webových prohlížečů, které se nedržejí specifikace a tím dochází při stejném kódu k různým vzhledům.

Kapitola 3

Analýza a návrh

V této kapitole je nejříve popsána specifikace požadavků a následně je znázorněna analýza a návrh webového portálu na základě těchto požadavků.

3.1 Specifikace požadavků

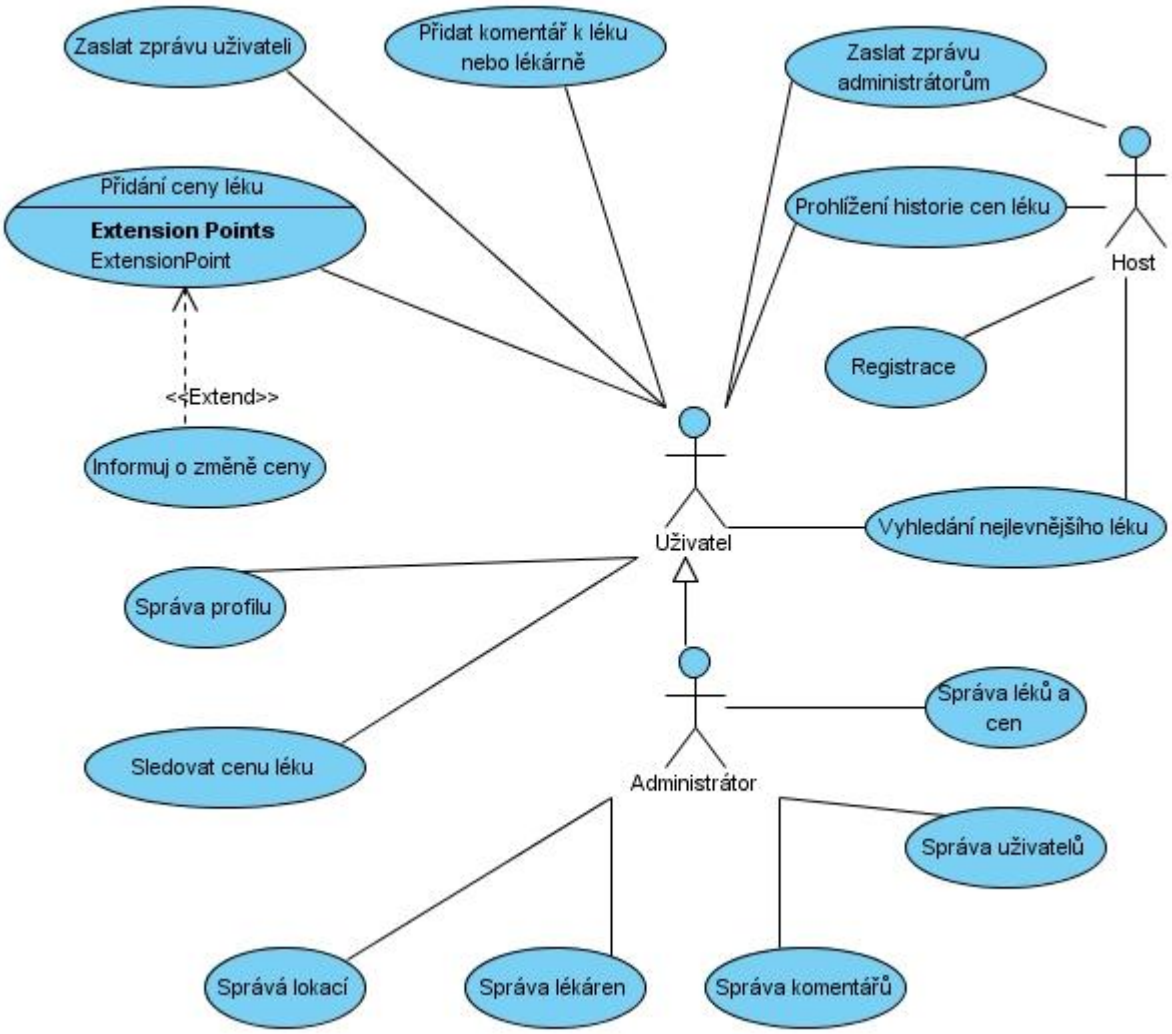
Hlavním cílem webového portálu je pomoci návštěvníkovi nalézt nejvýhodnější cenu léku v požadované lokaci. Ceny do systému zadávají registrovaní uživatelé, kteří mají dále výhodu v možnosti, nechat si zasílat oznámení o změně ceny u vybraného léku, přidávat komentáře nebo posílat jiným uživatelům soukromé zprávy. U každého léku by měla být možnost sledovat vývoj ceny. Správu lékáren a s tím související správu lokací provadí administrátor webového portálu, který se také stará o odstraňování nevhodných komentářů. Administrátor má taktéž na starost správu uživatelů, aby mohl například odstranit účet uživatele, který záměrně přidává nesprávné ceny k lékům.

3.2 Návrh požadované funkčnosti

Návrh požadované funkčnosti je prezentován pomocí Diagramu případu použití.

Aktéři:

- **Host**
  Aktér host je návštěvník webového portálu, který není zaregistrován a jeho možnosti jsou omezené, přesto má k dispozici nejdůležitější funkci a to vyhledání ceny léku.
- **Uživatel**
  Uživatel je takový aktér, který má rozšířené možnosti, jako přidávání cen k lékům, nechávat si zasílat upozornění o změně cen léků, přidávat komentáře a zasílat soukromé zprávy ostatním uživatelům.
- **Administrátor**
  Administrátor je aktér s nejvyššími pravomocemi. Přidává do systému nové lékárny a léky. Spravuje ceny léků, samotné uživatele a stará se o mazání nevhodných komentářů. Má všechny práva, které má i aktér Uživatel.



Obrázek 3.1: Diagram případu užití

Případy užití:

- **Vyhledání nejlevnějšího léku**
  Aktérum je umožněno vyhledat nejnižší cenu léku podle zvolené lokace. Lokací je myšleno kraj, okres, obec či PSČ. Po zadání hledaného léku je možno filtrovat podle lokací seznam lékáren poskytující lék seřazený vzestupně podle ceny.

- **Prohlížení historie cen léku**
  Tento případ užití zprostředkovává aktérům historii cen léků. Je možné sledovat vývoj ceny a uživatele, kteří ceny zadali.

- **Registrace**
  Pokud si aktér Host přeje využívat rozšířené funkce portálu, je mu poskytnuta možnost se zaregistrovat.

- **Zaslat zprávu uživateli**
  Umožňuje přihlášenému uživateli zaslat soukromou zprávu v rámci systému jinému registrovanému uživateli. Uživatelé si tak můžou sdělovat informace o jednotlivých lécích.

- **Přidat komentář k léku nebo lékárně**
  Aktéři mají možnost k jednotlivým lékům nebo lékárnám přidávat své komentáře. Mohou se tak s ostatními podělit o zkušenosti s personálem lékáren nebo upozornit na vyjímečnou změnu otevírací doby.

- **Zaslat zprávu administrátorům**
  Dovoluje všem aktérům zaslat zprávu administrátorům. Například informovat o lékárně či léku, který chybí v naší databáze nebo zasílat připomínky k samotnému portálu.

- **Přidání ceny léku**
  Přihlášený uživatel zadává do systému aktuálnější cenu léku. Stávající cena se nepřepisuje z důvodu zachování historie cen.

- **Informuj o změně ceny**
  Rozširuje stávající funkčnost případu užití „Přidání ceny léku“ o zaslání informace daným uživatelům, že cena byla změněna.

- **Správa profilu**
  Umožňuje aktérovi měnit jeho uživatelské údaje, přednastavovat filtr a spravovat sledované ceny léků.

- **Sledovat cenu léku**
  Aktér u vybraného léku povolí službu sledování cen léku, která mu při změně ceny v jím nastavené lokalitě, zašle emailem upozornění.

- **Správa lokací**
  Lékárny jsou umístěny v určitých lokacích, které jsou v systému rozděleny na jednotlivé kraje, okresy, obce a PSČ.

- **Správa lékáren**
  Pouze administrátor má možnost spravovat lékárny. Lékárna je přiřazena do patřičné lokace a jsou o ní vyplněny požadované informace.

- **Správa uživatelů**
  Správa uživatelů poskytuje operace s uživateli. Nabízí možnost smazat účet nevhodně se chovajícím uživatelům či nastavit uživatele jako administrátora portálu.

- **Správa léku a cen**
  Poskytuje administrátorovi možnost spravovat léky a jejich ceny. Léky je též možné hromadně naimportovat.

- **Správa komentářů**
  Poskytuje administrátorům volbu smazání komentářů. Tato volba je určena pro účely mazání nevhodných komentářů.

Případ užití: Přidání ceny léku

Následující tabulka detailně popisuje případ užití Přidání ceny léku. Případ užití je jednoznačně identifikován svým ID a jsou definovány účastníci tohoto případu užití. Stěžejní částí je Tok událostí, která specifikuje tok operací, které se provádějí.

Případ užití: Přidání ceny léku
ID: UC03
Účastníci:
Uživatel nebo Administrátor
Vstupní podmínky:
Účastník přihlášen do systému
Tok událostí:
1. Účastník hledá lék.
2. POKUD lék nenalezl:
2.1 Nahlásí neexistující lék a jeho cenu.
2.2 Případ užití končí.
3. Uloží novou cenu léku.
4. Proveden případ užití "Informuj o změně ceny".
Alternativní tok 1:
Účastník se odhlásí ze systému.

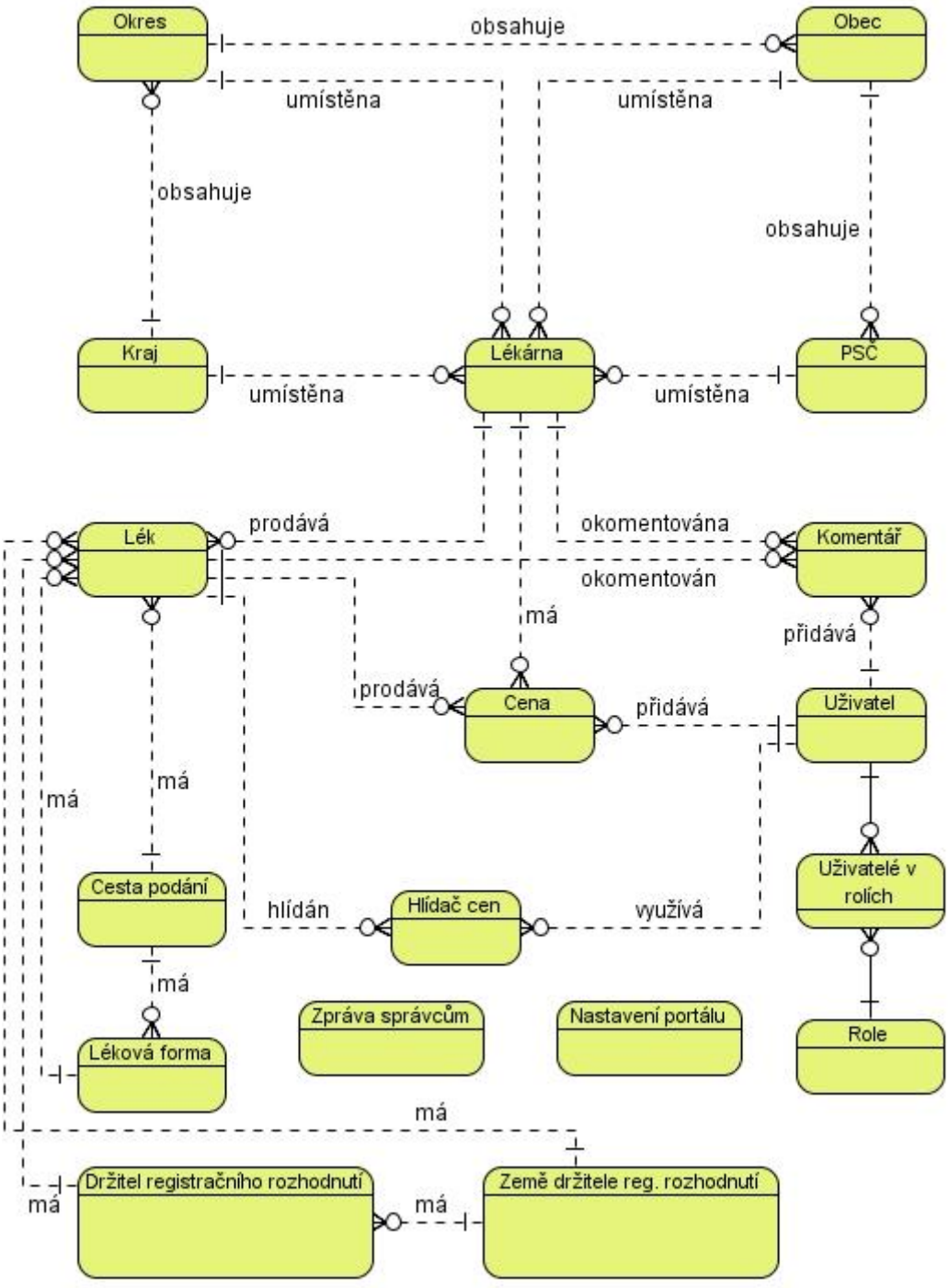
Obrázek 3.2: Případ užití: Přidání ceny léku

3.3 Konceptuální model

Konceptuální model vyjadřuje objekty působící v systému a jejich vazby. Poskytuje základ pro vytvoření databáze. Jednotlivé atributy entit byly vynechány, jelikož jsou patrné z Diagramu databáze, viz. přílohy.

Popis jednotlivých entit:

- **Kraj**
  Entita reprezentující kraj České republiky
- **Okres**
  Entita reprezentující okres, který vždy přísluší určitému kraji.
- **Obec**
  Entita reprezentující obec, která vždy přísluší určitému okresu.
- **PSČ**
  Entita reprezentující PSČ, které vždy přísluší určité obci.
- **Lékárna**
  Představuje entitu lékárny, která v sobě uchovává lokalitu umístění lékárny. Entita také obsahuje doplňující informace jako otevírací dobu lékárny.



Obrázek 3.3: Konceptuální model

- **Lék**
  Jedna z nejdůležitějších entit v aplikaci. Udržuje veškeré informace o léku, díky kterým mohou uživatelé přesněji vybrat hledaný lék.
- **Cesta podání**
  Vyjadřuje cestu podání, kterou je lék podáván.
- **Léková forma**
  Vyjadřuje formu, v které je lék podáván.
- **Země držitele reg. rozhodnutí**
  Entita představující zemi sídla držitele registračního rozhodnutí.
- **Držitel registračního rozhodnutí**
  Reprezentuje držitele registračního rozhodnutí.
- **Zpráva správcům**
  Kontaktní zpráva určená pouze pro správce aplikace. Zpětný kontakt na odesílatele je pouze emailová adresa, protože uživatel nemusí být registrován v aplikaci.
- **Nastavení portálu**
  Udržuje informace o nastavení portálu, které si mohou administrátoři libovolně měnit.
- **Uživatel**
  Entita představující registrovaného uživatele aplikace, který může přidávat komentáře, ceny a nastavovat sledování cen léků. Uživatel je svázán s entity lokací, které slouží pro předvolbu uživatelského filtru.
- **Role**
  Vyjadřuje roli uživatele v aplikaci, podle které jsou nastavována práva.
- **Uživatelé v rolích**
  Entita zajišťující vazbu N:N mezi entity Uživatel a Role. Definuje role přiřazené uživateli.
- **Komentář**
  Entita reprezentující komentář přidaný uživatelem k vybrané entitě léku nebo lékárně.
- **Cena**
  Představuje cenu léku v lékárně zadanou uživatelem. Cena může být dvojího typu, buď pouze doplatek z celkové částky nebo celá částka. V entitě je uchován datum přidání ceny a příznak, zda je cena aktuální.
- **Hlídač cen**
  Vyjadřuje entitu nastavení pro sledovaný lék, kterou si nastavuje uživatel. Entita má vazbu na lokace, ve kterých si uživatel přeje sledovat cenu léku.

Poznámka k obrázku 3.3:
Entity Hlídač cen a Uživatel mají stejnou vazbu na entity lokací(Kraj, Okres, Obec, PSČ) jako entita Lékárna. Vazby byly v konceptuálním modulu vynechány z důvodu zachování přehlednosti modelu.

3.4 Diagram tříd

Diagram tříd představuje logický pohled na aplikační logiku systému. Není shodný s datovým návrhem databáze, avšak umožnuje o ní získat určité poznatky. Diagram byl vygenerován z Microsoft Visual Studia, čímž je dosaženo, že diagram odpovídá vytvořenému systému.
Příloha: Diagram tříd

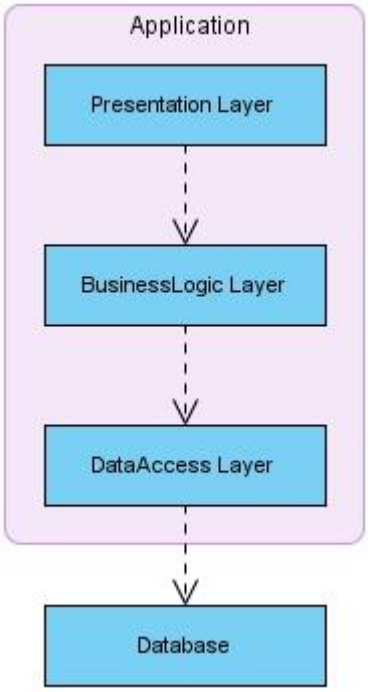
Kapitola 4

Implementace

Tato kapitola popisuje implementaci webového portálu. Podrobněji se zaměří na architekturu aplikace, zabezpečení, uživatelské účty a některé prvky samotného portálu.

4.1 Architektura aplikace

Jako architekturu aplikace jsem zvolil 3-vrstvou architekturu. Tato architekura logicky rozděluje aplikaci na vrstvy, které spolu podle určitých podmínek a pravidel komunikují. Jednotlivé vrstvy jsou popsány níže. Síla této architektury je zřejmá až ve větších projektech, kde mezi hlavní výhody patří spolupráce programátorů na úrovni vrstev, snadnější údržba a nezávislost aplikace na datovém úložišti. Celá aplikace je uzavřena do jmenného prostoru *BachelorsThesis*.



Obrázek 4.1: Achitektura aplikace

4.1.1 Databáze

Při implementace databáze se vycházelo z konceptuálního modelu z předchozí kapitoly. Všechny data jsou uložena do tabulek a každá tabulka má svůj primární klíč, který identifikuje jedinečný záznam. Relace mezi tabulkami jsou implementovány pomocí cizích klíčů. Pro všechny datové operace, které se budou z webového portálu provádět, jsou připraveny uložené procedury.

Cizí klíče mají možnost vyvolávat akci kaskádovitého mazání záznamů. Tato volba ušetřuje část práce, ale nelze ji využívat všude, protože při komplikovanějších relacích by docházelo k cyklení. Na tabulkách, kde jsem nemohl využít kaskádovitého mazání, jsem využil databázové triggery, které už jsou náročnější na implementaci, avšak stále poskytují poměrně jednoduché řešení mazání záznamů.
Příloha: Diagram databáze

4.1.2 DataAccess Layer

Vrstva přístupu dat provádí operace nad databází. V tomto případě vždy volá uložené procedury s odpovídajícími vstupními a výstupními parametry. Výsledky operací se vrací do vyšší vrstvy, která operaci zavolala. Výhodou takto oddělené vrstvy přístupu dat je v tom, že pokud by se do budoucna měnilo datové úložiště, např. na databázi Oracle, tak stačí pouze vyměnit v aplikaci tuto vrstvu se zachovaným rozhraním a zbytek aplikace může zůstat nezměněn. Vrstva přístupu dat je užavřena do jmenného prostoru *BachelorsThesis.DataAccess*.

Všechny třídy operující s databází s aplikačními objekty jsou potomky abstraktní třídy *DataAccessBase*, která obsahuje obecné metody pro ukládání a mazání aplikačních objektů v databázi. Ve vrstvě je dále statická třída *StoredProc*, která uchovává na jediném místě názvy všech uložených procedur v databázi.

Pro připojení do databáze je potřeba připojovací řetězec, ve kterém se definuje databázový server, autorizace do databáze a jiné parametry. Tento řetezec je použit při každém otevření spojení do databáze a proto je pro jeho správu lepší ho uložit na jediném místě. V tomto případě je nevhodnějším místem konfigurační soubor webové aplikace web.config. Připojovací řetezec ve vrstvě zprostředkovává vlastnost ConnectionString třídy *DataConfig*.

```
<appSettings>
    <add key="ConnectionStringName" value="Development"/>
</appSettings>
<connectionStrings>
    <clear/>
    <add name="Development"
         connectionString="Data Source=A04-A0511; Initial Catalog=BC; user id=dbUser; password=heslo123"/>
    <add name="Production"
         connectionString="Data Source=192.168.1.6; Initial Catalog=db261; user id=db261; password=heslo123;"/>
</connectionStrings>
```

Obrázek 4.2: Nastavení připojovacího řetězce

4.1.3 Business Layer

Vrstva obsahuje aplikační objekty a logiku aplikace. Je prostředníkem mezi prezentační vrstvou a vrstvou přístupu dat, díky čemuž je možné mít implementovanou prezentační vrstvu zcela nezávislou na datovém úložišti. V této vrstvě by také bylo možné provádět validace vlastností aplikačních objektů při ukládání dat, ale v případě této práce jsou validace prováděny v prezentační vrstvě.

BusinessObjects

Aplikační objekty představují třídy, které reprezentují objekty figurující v aplikaci. Tyto aplikační objekty byly záměrně odděleny od aplikační logiky, aby bylo možné jejich využití ve vrstvě přístupu dat. Toto oddělení bylo provedeno z důvodu cyklení při kompilaci aplikace ve vývojovém prostředím Visual Studio, ke kterému docházelo při nastavení propojení mezi aplikační logikou a vrstvou přístupu dat.

Všechny aplikační objekty jsou potomky abstraktní třídy *BusinessObject*, ve které jsou definovány společné vlastnosti pro všechny objekty. V současnosti je to pouze identifikátor objektu, avšak tato abstraktní třída může v budoucnu ušetřit práci při úpravách či rožšířeních aplikace.

Tato vrstva je uzavřena ve jmenném prostoru *BachelorsThesis.BusinessObjects*, ve kterém je ještě uzavřen jmenný prostor *BachelorsThesis.BusinessObjects.Collections*. V tomto prostoru jsou umístěny kolekce nad objekty pro jednodušší zacházení s celou sadou objektů. Tyto kolekce jsou potomky typu Collection daného aplikačního objektu.

BusinessLogic

Aplikační logika obsahuje třídy, které provádějí mimo jiné operace čtení, ukládání a mazání nad aplikačními objekty. Metody provádějící tyto operace mají v sobě část aplikační logiky a volají odpovídající metody ve vrstvě přístupu dat. Pro lepší orientaci při vývoji aplikace, mají tyto třídy shodný název jako objekt, nad kterým pracují, se sufixem *Manager*. Pro snažší práci s třídami v prezentační vrstvě jsou třídám a jejím vybraným metodám přidány parametry, které označují objekt vhodný pro vázání na ObjectDataSource.

Abych využil možnosti stránkování ovládacích prvků pro prohlížení více objektů v prezentační vrstvě, musel jsem metody pro získávání kolekce objektů přetížit. Toto přetížení spočívalo v přidání parametrů pro stránkování, jako index stránky a její velikost. S tím souviselo i přidání metod pro získání počtu přečtených záznamů, které jsou pro správnou funkčnost stránkování nutné. Stránkování na úrovni webového serveru znamená načtení celé sady záznamů z databáze, což je operace, která by se brzy podepsala na výkonu portálu. Proto jsem zavedl stránkování na úrovni databáze, s čímž souvisely změny ve vrstvě přístupu dat a v uložených procedurách.

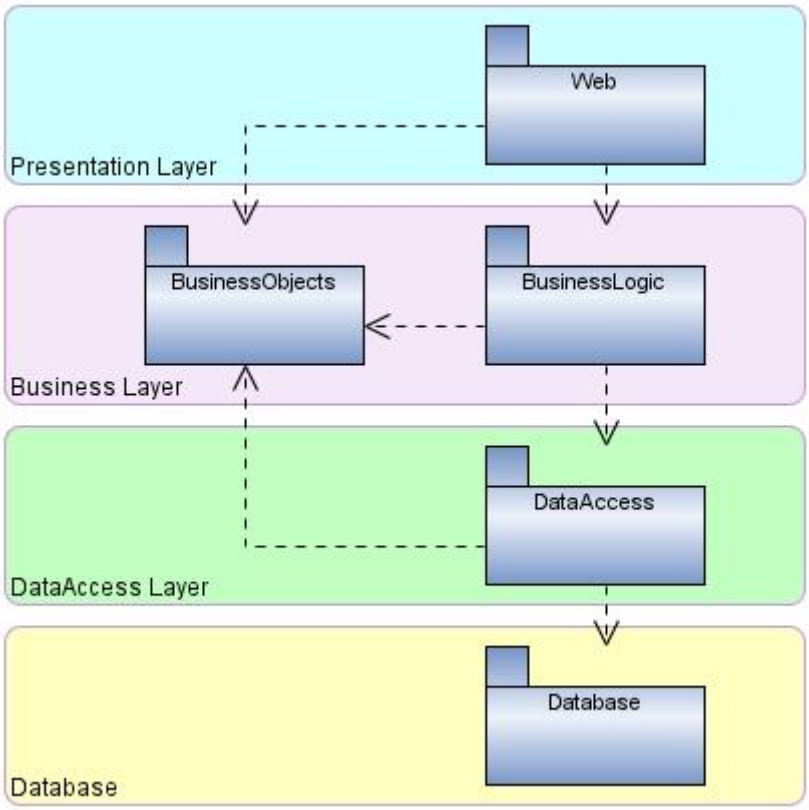
V aplikační logice jsou i třídy, které neprovádějí operace nad aplikačními obejkty. Jsou to třídy, jež provádějí operace, které nejsou závislé na prezentační vrstvě a mohou být umístěny v aplikační logice, jako například operace zasílání oznámení uživatelům o změně ceny.

Aplikační logika je uzavřena do jmenného prostoru *BachelorsThesis.BusinessLogic*, do které je ješte uzavřena aplikační logika importu dat léků. Import dat je uzavřen do jmenného prostoru *BachelorsThesis.BusinessLogic.Import*. Import bude popsán podrobněji dále v této kapitole.

4.1.4 Presentation Layer

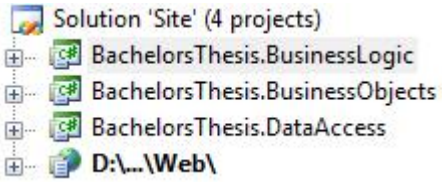
Prezentační vrstva je nejvyšší vrstva, přes kterou uživatel komunikuje se systémem. V tomto případě je použita vrstva ASP.NET WebForms pro webové aplikace, ale může být nahrazena např. vrstvou WinForms pro desktopové aplikace beze změny v nižších vrstvách. Vrstva nekomunikuje přímo s vrstvou přístupu dat, ale využívá prostředníka v podobě vrstvy aplikační logiky, čímž se docílí nezávislosti na datovém zdroji.

Z výše uvedeného popisu jednotlivých vrstev byl vytvořen diagram balíčků, který znázorňuje vrstvy aplikace obsahující balíčky tříd.



Obrázek 4.3: Package diagram aplikace

Z tohoto diagramu balíčků jsem vycházel při zakládání projektu ve vývojovém prostředí Visual Studio.



Obrázek 4.4: Založený projekt ve Visual Studiu

4.2 Zabezpečení a uživatelské účty

ASP.NET poskytuje několik způsobů autentizace uživatelů. Pro tento webový portál byla nejvhodnější formulářová autentizace, kde se uživatel přihlašuje svým uživatelským jménem a heslem, které jsou společně uloženy v nějakém datovém úložišti.

4.2.1 API členství

Od verze ASP.NET 2.0 je možné využívat API členství (Membership). Je to rohraní, které implementovalo opakující se úkoly, které se využívají ve většině aplikací, jako je přihlašování a registrace uživatelů, změna hesla uživatele a jiné funkcionality týkající se administrace uživatelů. Členství také nabízí připravené ovládací prvky pro většinu z těchto funkcí.

API členství pracuje nezávisle na datovém úložišti. Pro přístup k datovému úložišti jsou určeny poskytovatelé členství, jež volají metody API členství. Poskytovatel členství implementuje přístup do datového úložiště a lze si napsat vlastní poskytovatele nad libovolným úložištěm. Právě tohoto jsem využil a použil jsem již hotového poskytovatele Altairis.Web.Providers.SimpleSqlMemebrshipProvider [4], který značně zjednodušil původní poskytovatele SQL serveru, jež byl pro účely portálu zbytečně komplexní a robustní.

```
<membership defaultProvider="MyMembershipProvider">
  <providers>
    <clear/>
    <add name="MyMembershipProvider"
         type="Altairis.Web.Security.SimpleSqlMembershipProvider"
         connectionStringName="Development"/>
  </providers>
</membership>
```

Obrázek 4.5: Nastavení poskytovatele členství

API členství poskytuje funkce pro:

- Vytváření a odstraňování uživatelů buď programátorsky nebo pomocí Web Administration Tool (WAT).
- Zasílání emailu uživateli žádajícímu o obnovu ztraceného hesla
- Vyhledávání uživatele v datovém úložišti nebo načtení celého seznamu pro získání ostatních informací o uživateli.
- Zabudování ovládacích prvků pro přihlašování a registraci uživatelů, zobrazení stavu přihlášení, obnovu hesla, různé režimy zobrazení pro přihlášené a anonymní uživatele.

4.2.2 API rolí

Stejně jako ASP.NET obsahuje rozhraní pro správu členství, tak také obsahuje rohraní pro správu rolí. Pomocí tohoto rohraní je možné spravovat role, přidělovat a odebírat role uživatelům a poskytuje také programátorský přístup k rolím. Správu rolí je možné provádět přes WAT.

Princip poskytovatele rolí funguje stejně jako u poskytovatele členství, proto jsem zde také využil zjednodušeného poskytovatele Altairis.Web.Providers.SimpleSqlRoleProvider [4].

```
<roleManager enabled="true" defaultProvider="MyRoleProvider">
  <providers>
    <clear/>
    <add name="MyRoleProvider"
         type="Altairis.Web.Security.SimpleSqlRoleProvider"
         connectionStringName="Development" />
  </providers>
</roleManager>
```

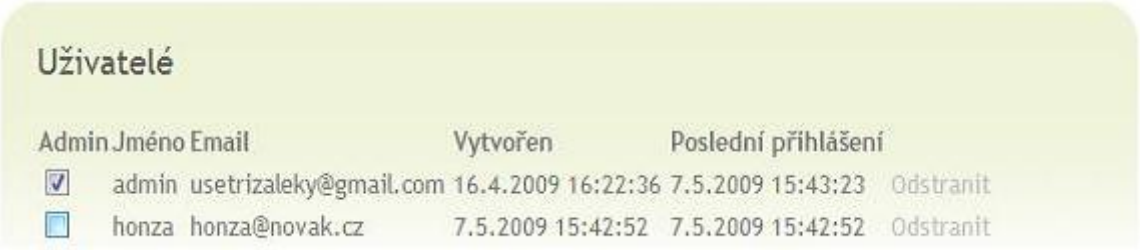
Obrázek 4.6: Nastavení poskytovatele rolí

Jak již bylo zmíněno v minulé kapitole, v systému se můžou objevit 3 druhy uživatelů. Nyní však popíšu pouze druhy uživatelů, kteří se mohou přihlásit do systému:

- **Uživatel**
  Tento druh uživatele není přiřazen do žádné role. Stránky, ke kterým má uživatel přístup, jsou v adresáři *Protected* a nastavení oprávnění je uloženo v konfiguračním souboru tohoto adresáře. Do tohoto adresáře mají přístup všichni přihlášení uživatelé, tedy včetně administrátora.

- **Administrátor**
  Uživatel typu Administrátor je přiřazen do role *admin*, která má jako jediná přístup do adresáře *Admin*, jež obsahuje stránky pro administraci portálu.

V aplikaci je připraveno rozhraní pro Administrátory, aby mohli přiřazovat vybraným uživatelům roli *admin*.



Obrázek 4.7: Správa uživatelů v aplikaci

4.3 Aktualizace cen

Jedním z hlavních pilířů portálu je funkcionalita aktualizace cen, která je kromě samotného přidání ceny léku rozšířena, ještě o zasílání uživatelům, kteří si sledování ceny léku nastavili, oznámení o změně ceny léku.

4.3.1 Přidání ceny léku

Přidávání ceny léku je založeno na lidské ochotě udělat něco pro ostatní. Možnost přidávat ceny léků mají pouze registrovaní uživatelé, jejiž IP adresa je ukládána do záznamu nové ceny. Toto ukládání IP adresy jsem zavedl z důvodu, že je zde možnost zneužití portálu ke konkurenčnímu boji lékáren tím, že k lékům v konkurenční lékárně by mohli uživatelé záměrně přidávat vyšší ceny. Pokud by bylo toto odhaleno, tak by bylo možné do budoucna těmto uživatelům zákázat přístup k portálu.

Přidávání cen se provádí stisknutím tlačítka ze stránky léku. Na stránce přidání ceny léku je uživatel povinnen vyplnit tři údaje. Prvním je vybrat lékárnu vyfiltrovanou podle lokality, ve které je možné lék za tuto cenu zakoupit. Dalším důležitým údajem je zvolit typ ceny. Na výběr je možnost *Doplatek*, což znamená částku, kterou zákazník doplácí k částce hrazenou pojišťovnou. Druhou možností je zvolit typ *Plná cena*, vyjadřující plnou částku, za kterou byl lék zakoupen. Posledním údajem je již samotná částka. Ceny se nepřepisují, ale přidávají. Díky tomu lze sledovat vývoj ceny léku v lékárně.

Obrázek 4.8: Přidání ceny léku

Na stránce jsou zabudovány validační prvky, které kontrolují zda byla cena zadána ve správném formátu, tj. například přítomnost čárky oddělující desetinnou část částky. Kontrolován je také správný výběr typu ceny a zda je vybrána lékárna. Validace jsou prováděny, jak na straně klienta pomocí Javascriptu, tak i na straně serveru např. pomocí vlastních validačních metod. Téměř celá stránka je umístěna v ovládacím prvku UpdatePanel ze sady nástrojů AJAX Extensions. Využití tohoto prvku má výhodu v tom, že stránka není vždy odesílána na server, ale operace jako filtrování lékáren a jejich výběr je prováděn asynchronní komunikací se serverem v pozadí stránky, takže uživatel není rušen neustálým znovu načítáním stránky.

4.3.2 Sledování ceny léku

Sledování ceny umožňuje uživateli být okamžitě informován při přidání ceny léku jiným uživatelem. Uživatel tedy při příštím nákupu ví, kde je daný lék nejlevnější. A aby se uživateli nezasílaly oznámení z celé republiky, tak má možnost nastavit si lokaci, pro kterou mu bude zasíláno oznámení o změně ceny. Tím pádem si může nastavit zasílání oznámení pouze pro část města, kde bydlí nebo třeba pro celé město. Oznámení je zasíláno na uživatelův email, který má vyplněný ve svém profilu.

Administrátorovi portálu je umožněno nastavení údajů odesílaného oznámení. Toto nastavení se provádí v Nastavení portálu v menu Administrace. Jako první údaj je nastavení emailu, z kterého se email odesílá. Samotná konfigurace pro zasílání emailu se nastavuje v konfiguračním souboru webové aplikace. Zde je možnost zapsaní jiného emailu, na který je přesměrováno z emailu v nastavení konfigurace. Dalším údajem je nastavení předmětu emailu. Posledním údajem je samotné tělo zprávy. V těle zprávy je povoleno přidávat HTML značky, takže lze vytvořit pro uživatele rychlý odkaz na aktualizovaný lék. Do těla zprávy je nutné přidat 2 pozice pro parametry, které identifikují lék, jehož cena byla aktualizována. Pozice první musí být v textu označena takto „{0}“ a při zpracování je na tuto pozici doplněn identifikátor léku v databázi. Pozice druhá musí být označena takto „{1}“ a je zde doplněn název léku.

Oznámení o změně ceny léku se zasílá uživatelům při přidání nové ceny. Jelikož lék může sledovat značný počet uživatelů, mohlo by být přidání ceny poměrně časově náročné a uživatele, který cenu přidal by tato akce pouze zdržovala a přitom by pro něj nenesla žádný užitek. Z tohoto důvodu jsem se rozhodl zasílat oznámení o změně ceny ve vlastním vláknu, takže uživatel ani nebude vědět, že v pozadí probíhá ještě nějaká náročná operace.

```
/// <summary>
/// Zašle oznámení uživatelům o nové ceně
/// </summary>
/// <param name="price">Cena</param>
public static void SendNotification(Price price)
{
    EmailPriceNotification notification = new EmailPriceNotification(price);
    ThreadStart threadStart = new ThreadStart(notification.SendNotification);
    Thread thread = new Thread(threadStart);
    thread.Start();
}
```

Obrázek 4.9: Zasílání oznámení pomocí vláken

4.4 Vyhledání léku a jeho ceny

Vyhledávání léku bude v portálu pravděpodobně jedna z nejvyužívanějších funkcionalit a proto jsem jí věnoval odpovídající pozornost. Jelikož jsem chtěl, aby hledání bylo pro uživatele co nejjednoduší a nejpříjemnější, rozhodl jsem se pro vyhledávaní využít fulltextové vyhledávání nad všemi daty týkajících se léků a souvisejících číselníků. Fulltextové vyhledávání není citlivé na velikost písmen hledaného řetězce a vrací výsledky i při zadaní jenom začátku slova. To znamená, že pokud uživatel hledá Paralen 500 a nechce se zdržovat psaním celého názvu, tak stačí například napsat „para“ a následně už si může vybrat z vrácených výsledků hledání.

Ceny léků lze prohlížet na stránce určitého léku. Je možné zde vyhledávat jak doplatek za lék, tak i jeho plnou cenu. Zobrazují se vyfiltrované lékárny podle lokace seřazené vzestupně podle ceny léku. Uživatel tedy vždy jako první vidí nejlevnější cenu, za kterou lze lék koupit v jím vybrané lokaci. Registrovaní uživatelé mají výhodu v tom, že si mohou přednastavit filtr lokací a nemusí ji při hledání ceny vždy vybírat.

Obrázek 4.10: Hledání nejnižší ceny léku

Aby uživatel nemusel vždy hledat svůj lék, implementoval jsem ukládání naposled prohlížených léků. Uživatel tedy při opětovné návštěvě portálu vídí, které léky naposled prohlížel a může se rychleji podívat na jejich ceny či nové komentáře. Prohlížené léky se ukládají do Cookies na klientské stanici a jejich počet je omezen na 5, poté se přemazávají naposled prohlížené.

4.5 Komentáře

Jelikož je tento portál založen na důveryhodnosti a ochotnosti uživatelů pomáhat lidem, kteří aktualizují ceny léků, rozhodl jsem se pro rozšíření aktivity na portálu implementovat k jednotlivým lékům a lékárnám možnost přidávat registrovaným uživatelům komentáře. Ty mohou být určeny k nejrůznějším účelům jako je například u lékáren možnost přidat komentář s otevírací dobou lékárny v době vánoc nebo dovoluje uživateli přidat komenář k léku, kde popisuje zda mu lék pomáhá. Využití komentářů je mnoho a záleží až na samotných uživatelých, jak je využijí. Bohužel je možné, že uživatelé budou přidávat nevhodné či nesouvisející komentáře, proto je administrátorovi aplikace uděleno právo jednotlivé komentáře mazat.

Protože jsou komentáře jak u léků, tak i u lékáren, snažil jsem se je implementovat tak, aby byly v aplikaci implementovány pouze jednou a nemusel se psát kód k lékům a lékárnám zvlášť. Abych tohoto docílil využil jsem možnosti ASP.NET a komentáře naimplementoval jako uživatelský ovládací prvek (Web User Control). To znamená, že veškerou funkcionalitu týkající se komentářů mám na jednom místě a pokud je potřeba je připojit k části aplikace, kde by byly komentáře vhodné, přidám na danou stránku tento ovládací prvek s nastavením, k čemu se komentář váže a identifikátor objektu. Toto řešení má výhody v tom, že budoucí úpravy nebo rozšíření se provádějí na jediném místě, ale výsledek bude vidět u všech použitých ovládacích prvků.

Obrázek 4.11: Komentáře

4.6 Soukromé zprávy

Jelikož je možné, že někteří uživatelé budou chtít mezi sebou komunikovat, tak byla v portálu implementována funkcionalita pro zasílání soukromých zpráv mezi registrovanými uživateli. Pro komunikaci uživatelů už jsou sice implementovány komentáře, avšak pokud by

chtěli uživatelé komunikovat soukromně, nemeli pro to žádnou možnost. U uživatelů je sice uložen email, ale rozhodl jsem se ho nezveřejňovat, aby nedošlo k jeho zneužití.

Zprávy lze zasílat ze stránky Zprávy v uživatelském menu. Z tohoto místa mohou uživatelé zasílat zprávy jiným uživatelům. Druhou možností, jak zaslat zprávu, je kliknout na jméno uživatele, při zobrazení cen léku nebo komentáře. Tím dojde k přesměrování na soukromé zprávy a zároveň k předvyplnění jména uživatele, kterému má být zpráva směrována. Při odesílání zprávy se provádějí validace zda jsou všechny údaje vyplněny a zda vyplněný příjemce v aplikaci opravdu existuje.

Na stránce soukromých zpráv uživatel vidí všechny přijaté zprávy, které mu ostatní uživatelé zaslali. Pro lepší orientaci v přijatých zprávách byl doplněn příznak, který identifikuje, zda byla už zpráva přečtena. Pomocí jQuery pluginu byly implementovány záložky, které umožní jednoduché přepnutí na všechny odeslané zprávy. I u odeslaných zpráv lze sledovat, zda byla již zpráva příjemcem přečtena. Uživatel, který zprávu odeslal, má tak možnost zjistit, jestli na jeho zprávu příjemce nechce odpovědět nebo si ji ještě nepřečetl.

Obrázek 4.12: Seznam přijatých zpráv

Při zobrazení přijaté zprávy se otevře stránka se všemi informacemi týkající se zprávy, včetně samotného obsahu.

Obrázek 4.13: Přijatá zpráva

4.7 Import dat

Rozhodl jsem se do portálu implementovat možnost importu dat léků. Tyto data poskytuje Státní ústav pro kontrolu léčiv (SUKL) [6]. Data jsou poskytována včetně všech číselníků, na které je navázána hlavní tabulka s léky. Původním záměrem bylo provádět import dat z XML souboru, ale jelikož SUKL poskytuje data ve formě databáze dBase uložené v souboru *.dbf, tak jsem se rozhodl provádět import z těchto souborů. Pro tuto volbu jsem se rozhodl, protože pokud bych zůstal u importu dat z XML, tak by se data musela nejdříve komplikovaně převádět do XML a až následně by je bylo možné importovat.

Do databáze portálu se neimportují všechny data, která poskytuje SUKL, ale pouze ty, která mají pro návštěvníka význam. Pro zachování možnosti zpětného dohledání v originálních datech, jsem do všech importovaných tabulek přidal navíc sloupec *SUKLId*, který vyjadřuje primární index z původní tabulky. Pokud už záznam s tímto identifikátorem existuje, tak jsou pouze aktualizovány jeho hodnoty. Záznam z importované tabulky, je importován pouze tehdy, pokud už byly naimportovány záznamy, na které je vázán. Tímto je zajištěna konzistentnost v importovaných tabulkách.

Pokud je importovaný záznam svázan s nějakým jiným záznamem, tak se ve vázané tabulce hledá záznam skrze sloupec *SUKLId*, aby se mohl uložit do importované tabulky nový identifikátor svázaného záznamu. Tato častá komunikace mezi databázovým a webovým serverem by mohla způsobit poměrně velké zpomalení importu. Proto jsem se rozhodl pro řešení s načtením kompletní tabulky do kolekce daných aplikačních objektů, ve které následně pomocí technologie LINQ provádím dotazy podobné SQL příkazům. Tímto by měla být zajištěna rychlost importu při importování většího množství dat.

Obrázek 4.14: Import dat

4.8 Logování chyb

Pro logování vyjímek v aplikaci a různých chyb jsem nasadil open source knihovnu ELMAH (Error Logging Modules And Handlers) [1]. Pomocí této knihovny lze snadno sledovat chyby vyvolané v aplikaci, které se objevily uživatelům. Jelikož řadový uživatelé nedokážou vždy úplně nejpřesněji popsat vyvstalý problém, tak tato knihovna zajistí vývojáři přesný popis chyby, díky kterému lze zapracovat na její opravě.

Knihovna umí kromě logování, také zasílání logované zprávy na vybraný email, aby bylo možné zajistit okamžitou nápravu. Logování chyb je možné do několika datových úložišť, např. SQL server nebo XML soubor. V případě tohoto portálu byla zvolen SQL server. Knihovna také podporuje filtrování chyb, takže je možné vypustit chyby, které nejsou relevantní a pouze by znepřehledňovaly log.

Velkou výhodou této knihovny je to, že poskytuje uživatelské rohraní pro sledování logovaných zpráv. To zajišťuje HTTP handler v konfiguračním souboru. Handler je nastaven na cestu do adresáře *Admin*, takže je zajištěno, aby logované zprávy mohl prohlížet pouze administrátor aplikace.

Veškeré nastavení se provádí v konfiguračním souboru webové aplikace web.config. Jak už jsem uvedl výše, jako datové úložiště byl zvolen SQL server a bylo nastaveno filtrování pro nelogování chybových zpráv s návratovým kódem 404, tj. stránka nebyla nalezena. Dále je také nutné nastavit HTTP modul, který se stará o odchytávání vyvstalých chyb a jejich logování. Na obrázku 4.16 je vidět upravený konfigurační soubor s nastavením knihovny ELMAH.

Error Log for /Web on A04-A0511

RSS FEED | RSS DIGEST | DOWNLOAD LOG | HELP | ABOUT

Errors 1 to 15 of total 68 (page 1 of 5). Start with 10, 15, 20, 25, 30, 50 or 100 errors per page.

Host	Code	Type	Error	User	Date	Time
A04-A0511	500	**InvalidCast**	Určené přetypování není platné. Details...	admin	2.5.2009	13:50
A04-A0511	500	**InvalidCast**	Určené přetypování není platné. Details...	admin	2.5.2009	13:47
A04-A0511	500	**Http**	Prvek GridView gvChemists aktivoval událost PageIndexChanging, která nebyla zpracována. Details...	admin	2.5.2009	13:11

Obrázek 4.15: Uživatelské prostředí ELMAH

```
<configSections>
    <sectionGroup name="elmah">
        <section name="errorLog" requirePermission="false"
                 type="Elmah.ErrorLogSectionHandler, Elmah" />
        <section name="errorFilter" requirePermission="false"
                 type="Elmah.ErrorFilterSectionHandler, Elmah"/>
        <section name="security" requirePermission="false"
                 type="Elmah.SecuritySectionHandler, Elmah"/>
    </sectionGroup>
</configSections>
<elmah>
    <errorLog type="Elmah.SqlErrorLog, Elmah"
              connectionStringName="Development" />
    <errorFilter>
        <test>
            <equal binding="HttpStatusCode" value="404" valueType="Int32"/>
        </test>
    </errorFilter>
    <security allowRemoteAccess="1" />
</elmah>
<system.web>
    <httpHandlers>
        <add verb="POST,GET,HEAD" path="Admin/elmah.axd"
             type="Elmah.ErrorLogPageFactory, Elmah" />
    </httpHandlers>
    <httpModules>
        <add name="ErrorLog" type="Elmah.ErrorLogModule, Elmah"/>
    </httpModules>
</system.web>
```

Obrázek 4.16: Nastavení knihovny ELMAH

Kapitola 5

Závěr

Úkolem této bakalářské práce bylo vytvořit webový portál doplatků léků a lékáren. Po nastudování problematiky tvorby cen léků, jsem pochopil význam tohoto portálu a jeho využití běžnými uživateli. S tímto vědomím jsem brzy vědel, jak by měl portál přibližně fungovat a co by měl obsahovat. Před samotnou implementací však bylo nutné provést analýzu a návrh aplikace, aby se předešlo budoucím problémům. Čas strávený analýzou a návrhem se vyplatil, jelikož jsem už prakticky nemusel upravovat původní návrh. Pouze jsem přidal nové prvky, které rozšiřovaly funkcionalitu aplikace o komunitní prvky jako jsou komentáře a soukromé zprávy. Toto rozšíření nezpůsobylo žádnou změnu v původním návrhu, takže jeho doplnění bylo poměrně jednoduché.

Jedním z problémů, s kterými jsem se musel vypořádat, byl import dat. Data, která poskytuje Státní ústav pro kontrolu léčiv, jsou někdy neúplná a import do „ostré“ aplikace, bez opravení poskytovaných dat, by nebyl nejvhodnějším řešením.

Díky vývoji portálu jsem se konečně blíže seznámil s vícevrstvou architektou, se kterou jsem dříve měl mizivou zkušenost. Dále jsem si rozšířil zkušenosti s jazykem C# a poznal zblízka technologii ASP.NET. Při pohledu na vývoj těchto technoligií a velkou komunitu vývojářů věřím, že tyto technologie mají budoucnost a rád bych u nich zůstal a dále se zdokonaloval.

5.1 Vylepšení portálu

Možným rozšířením by bylo, jak už jsem se dříve zmínil, zablokování přístupu k portálu určitým návštěvníkům. Toto blokování by se provádělo na základě IP adresy. Znamenalo by to i zablokování uživatelů, kteří mají společnou IP adresu s zablokovaným uživatelem. Bohužel by to bylo pro zachování korektnosti portálu nutné.

Pro reálné využití portálu by bylo vhodné vyměnit odkazy s identifikátory na jednotlivé entity za „hezčí“ pomocí metody url rewriting. Toto by se provádělo z důvodu optimalizace pro vyhledávače, aby uživatelé, kteří portál ještě neznají, ho lépe našli. Těchto technik je více, tuto jsem však uvedl pouze na ukázku.

Důležitou věcí je také odstranění chyb. Některé chyby se objeví až při reálném využití, proto je implementováno logování chyb, které napomůže s jejich nalezením a opravou.

Literatura

[1] Atif Aziz: Error Logging Modules and Handlers for ASP.NET [online]. 2009 [cit. 2009-05-18].
URL http://code.google.com/p/elmah/

[2] Hana Kanisová, Miroslav Müller: *UML srozumitelně*. Computer Press, 2006, ISBN 80-251-1083-4.

[3] Matthew MacDonald: *ASP.NET 3.5 a C# 2008*. Zoner Press, 2008, ISBN 978-80-7413-008-3.

[4] Michal Valášek: Altairis Simple ASP.NET SQL Providers [online]. 2000-2008 [cit. 2009-05-18].
URL http://www.aspnet.cz/Articles/115-altairis-simple-asp-net-sql-providers-ke-stazeni.aspx

[5] OBOUT SOFTWARE INC: Introduction to the .NET Framework [online]. 2009 [cit. 2009-05-18].
URL http://www.obout.com/show/show_slideshow.aspx

[6] Státní ústav pro kontrolu léčiv: Státní ústav pro kontrolu léčiv [online]. 2007 [cit. 2009-05-18].
URL http://www.sukl.cz

Dodatek A

Seznam příloh

Příloha 1. Diagram databáze - na přiloženém CD v `Prilohy/dbDiagram.jpg`
Příloha 2. Diagram tříd - na přiloženém CD v `Prilohy/classDiagram.jpg`

Dodatek B

Obsah CD

`xkodes00.pdf`	elektronická verze této práce
`Prilohy/`	adresář obsahující přílohy
`Site/`	kořenový adresář zdrojových kódů
`Site/BachelorsThesis.BusinessLogic/`	adresář obsahující třídy aplikační logiky
`Site/BachelorsThesis.BusinessLogic/Import/`	adresář obsahující třídy aplikační logiky importu
`Site/BachelorsThesis.BusinessObjects/`	adresář obsahující třídy aplikačních objektů
`Site/BachelorsThesis.BusinessObjects/Collections/`	adresář obsahující třídy kolekcí aplikačních objektů
`Site/BachelorsThesis.BusinessObjects/Enums/`	adresář obsahující výčtové typy
`Site/BachelorsThesis.DataAccess/`	adresář obsahující třídy vrstvy přístupu dat
`Site/Web/`	adresář obsahující prezentační vrstvu aplikace
`Site/Web/Admin/`	obsahuje stránky a nastavení pro administrátory
`Site/Web/App_Code/`	obsahuje třídy logiky v prezentační vrstvě
`Site/Web/App_Themes/`	obsahuje soubory definující vzhled portálu
`Site/Web/Bin/`	obsahuje knihovny nutné pro běh portálu
`Site/Web/Controls/`	obsahuje vlastní ovládací prvky
`Site/Web/Images/`	obsahuje obrázky využité v portálu
`Site/Web/js/`	obsahuje soubory javascriptu
`Site/Web/MasterPages/`	obsahuje vzory stránek portálu
`Site/Web/Protected/`	obsahuje stránky přístupné pouze registrovaným uživatelům
`Site/Web/SQL scripts/`	obsahuje sql skript pro vytvoření databáze